PORTAL
THE ACM DIGITAL LIBRARY

> home > about > feedback > login

US Patent & Trademark Office

Try the *new* Portal design
Give us your opinion after using it.

Search Results

Search Results for: **[transformational AND grammars AND languages AND compilers ]**
Found **1,330** of **127,944 searched.**
**Warning: Maximum** result set of **200 exceeded. Consider refining.**

Search within Results

> Advanced Search

> Search Help/Tips

**Sort by:** **Title** **Publication** Publication Date **Score**

**Results 1 - 20 of 200** short listing

Prev Page 1 2 3 4 5 6 7 8 9 10 Next Page

**1** A truly generative semantics-directed compiler generator 100%
Harald Ganzinger , Robert Giegerich , Ulrich Möncke , Reinhard Wilhelm
**ACM SIGPLAN Notices , Proceedings of the 1982 SIGPLAN symposium on Compiler construction** June 1982
Volume 17 Issue 6

This paper describes semantic processing in the compiler generating system MUG2. MUG2 accepts high-level descriptions of the semantics of a programming language including full runtime semantics, data flow analysis, and optimizing transformations. This distinguishes MUG2 from systems such as YACC [Joh75], HLP [HLP78], PQCC [PQC79], or its own former version [GRW77] with respect to expressive power and convenience. In this respect, MUG2 comes close to semantics-directed systems such as [Mos76 ...

**2** Generation of LR parsers by partial evaluation 98%
Michael Sperber , Peter Thiemann
**ACM Transactions on Programming Languages and Systems (TOPLAS)** March 2000
Volume 22 Issue 2

The combination of modern programming languages and partial evaluation yields new approaches to old problems. In particular, the combination of functional programming and partial evaluation can turn a general parser into a parser generator. We use an inherently functional approach to implement general LR(k) parsers and specialize them with respect to the input grammars using offline partial evaluation. The functional specification of LR parsing yields a concise implementat ...

**3** Attribute coupled grammars 97%

Harald Ganzinger , Robert Giegerich
**ACM SIGPLAN Notices , Proceedings of the 1984 SIGPLAN symposium on Compiler construction** June 1984
Volume 19 Issue 6

In this paper, attribute grammars are viewed as specifying translations from source language terms into target language terms. The terms are constructed over a hierarchical signature consisting of a semantic and a syntactic part. Attribute grammars are redefined to become morphisms in the category of such signatures, called attribute coupled grammars, such that they come with an associative composition operation. The composition allows for a new kind of modularity in compiler specifications. The ...

**4** Fast detection of communication patterns in distributed executions 97%
Thomas Kunz , Michiel F. H. Seuren
**Proceedings of the 1997** conference of the Centre **for Advanced Studies on Collaborative research** November 1997

Understanding distributed applications is a tedious and difficult task. Visualizations based on process-time diagrams are often used to obtain a better understanding of the execution of the application. The visualization tool we use is Poet, an event tracer developed at the University of Waterloo. However, these diagrams are often very complex and do not provide the user with the desired overview of the application. In our experience, such tools display repeated occurrences of non-trivial commun ...

**5** Experience with a Graham-Glanville style code generator 97%
Philippe Aigrain , Susan L. Graham , Robert R. Henry , Marshall Kirk McKusick , Eduardo Pelegri-Llopart
**ACM SIGPLAN Notices , Proceedings of the 1984 SIGPLAN symposium on Compiler construction** June 1984
Volume 19 Issue 6

**6** Modular specification of incremental program transformation systems 96%
Alan Carle , Lori Pollock
**Proceedings of the 11th** international conference **on Software engineering** May 1989

**7** Graph rewrite systems for program optimization 96%
Uwe Assmann
**ACM Transactions on Programming** Languages and **Systems (TOPLAS)** July 2000
Volume 22 Issue 4

Graph rewrite systems can be used to specify and generate program optimizations. For termination of the systems several rule-based criteria are developed, defining exhaustive graph rewrite systems. For nondeterministic systems stratification is introduced which automatically selects single normal forms. To illustrate how far the methodology reaches, parts of the lazy code motion optimization are specified. The resulting graph rewrite system classes can be e ...

**8** Tree transformation techniques and experiences 96%
S. E. Keller , J. A. Perkins , T. F. Payton , S. P. Mardinly
**ACM SIGPLAN Notices , Proceedings of the 1984 SIGPLAN symposium on Compiler construction** June 1984
Volume 19 Issue 6

A formal description technique for describing transformations from one well-defined language to another is introduced. A TT-grammar contains context-free grammars for describing the syntax of both languages. The transformation between the languages is described by a relationship of productions from the grammars. The TT-grammar is supported by an automated tool. SSAGS -- a translator writing system based on attribute grammars -- has been extended to support certain classes of TT-grammars. SSAGS a ...

**9** Handling Operator Precedence in Arithmetic Expressions with Tree Transformations 95%
Wilf R. LaLonde , Jim des Rivieres
**ACM Transactions on Programming Languages and Systems (TOPLAS)** January 1981
Volume 3 Issue 1

**10** Design, implementation and evaluation of the FNC-2 attribute grammar system 95%
Martin Jourdan , Didier Parigot , Catherine Julié , Olivier Durin , Carole Le Bellec
**ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1990 conference on Programming language** design and implementation June 1990
Volume 25 Issue 6

FNC-2 is a new attribute grammar processing system aiming at expressive power, efficiency, ease of use and versatility. Its development at INRIA started in 1986, and a first running prototype is available since early 1989. Its most important features are: efficient exhaustive and incremental visit-sequence-based evaluation of strongly (absolutely) non-circular AGs; extensive space optimizations; a specially-designed AG-description language, with provisions for true modularity; portability a ...

**11** Parse tree annotations 95%
James J. Purtilo , John R. Callahan
**Communications of the ACM** December 1989
Volume 32 Issue 12

A technique for associating rewrite rules with productions so that many high-level transformations of a source file can be generated easily is described. While eclipsed in power by other editing and compiler generation systems supporting management of both synthesized and inherited attributes, this approach is especially simple to employ and is sufficient in power to deal with a wide class of problems arising from practical applications.

**12** Composing tree attributions 95%
John Boyland , Susan L. Graham
**Proceedings of the 21st** ACM SIGPLAN-SIGACT symp**osium on Principles of programming languages** February 1994

Using the simple tree attributions described in this paper, attribute values can themselves be trees, enabling attribution to be used for tree transformations. Unlike higher-order attribute grammars, simple tree attributions have the property of descriptional composition, which allows a complex transformation to be built up from simpler ones, yet be executed efficiently. In contrast to other formalisms that admit descriptional composition, notably composable ...

**13** A slicing-based approach for locating type errors 95%
F. Tip , T. B. Dinesh

**ACM Transactions on Software Engineering and Methodology (TOSEM)** January 2001
Volume 10 Issue 1

The effectiveness of a type-checking tool strongly depends on the accuracy of the positional information that is associated with type errors. We present an approach where the location associated with an error message e is defined as a slice Pe of the program P being type-checked. We show that this approach yields highly accurate positional information: Pe is a progr ...

**14** A general approach for run-time specialization and its application to C 95%

Charles Consel , François Noël
**Proceedings of the 23rd ACM SIGPLAN-SIGACT symposium on Principles of programming languages** January 1996

**15** Composable attribute grammars: support for modularity in translator design and implementation 94%

R. Farrow , T. J. Marlowe , D. M. Yellin
**Proceedings of the 19th ACM SIGPLAN-SIGACT symposium on Principles of programming languages** February 1992

This paper introduces Composable Attribute Grammars (CAGs), a formalism that extends classical attribute grammars to allow for the modular composition of translation specifications and of translators. CAGs bring to complex translator writing systems the same benefits of modularity found in modern programming languages, including comprehensibility, reusability, and incremental meta-compilation. A CAG is built from several smaller component AGs, each of which solve ...

**16** Separating binding times in language specifications 94%

Torben Æ Mogensen
**Proceedings of the fourth international conference on Functional programming languages and computer architecture** November 1990

**17** An introduction to partial evaluation 94%

Neil D. Jones
**ACM Computing Surveys (CSUR)** September 1996
Volume 28 Issue 3

Partial evaluation provides a unifying paradigm for a broad spectrum of work in program optimization compiling interpretation and the generation of automatic program generators [Bjørner et al. 1987; Ershov 1992; and Jones et al. 1993]. It is a program optimization technique, perhaps better called program specialization, closely related to but different from Jørring and Scherlis' staging transformations [1986]. It emphasizes, in comparison with ...

**18** A globalizing transformation for attribute grammars 94%

K. J. Räihä , Jorma Tarhio
**ACM SIGPLAN Notices** , Proceedings of the 1986 SIGPLAN **symposium on Compiler contruction** July 1986
Volume 21 Issue 7

A transformation is presented for replacing conventional local attribute references in attribute grammars by global references. The purpose of the transformation is to enhance readability of the grammar and to facilitate easy storage optimization.

**19** Generation of formatters for context-free languages 94%

Mark van den Brand , Eelco Visser
**ACM Transactions on** Software Engineering and Method**ology (TOSEM)** January 1996
Volume 5 Issue 1

Good documentation is important [illegible] production of reusable and maintainable software. For the production of [illegible] documentation it is necessary that the original program text is not [illegible] manually to obtain a typeset version. Apart from being tedious, this will invariably introduce errors. The production of tools that support the production of legible and accurate documentation is a software engineering challenge in itself. We present an algebraic approach to the generation of tools ...

**20** Engineering A Program Optimizer 94%

John H. Crawford , Mehdi Jazayeri
**Proceedings of the 1978** annual conference December 1978

We describe our work in [illegible] an extension to an existing, large software system. In particular [illegible] in adding a global optimization phase to an operational [illegible] the module decomposition of the optimizer and how [illegible] specified. The resulting modules constitute a set of tools [illegible] promote the rapid and efficient implementation of porgram [illegible]. The information hiding strategy of Parnas w ...

**Results 1 - 20 of 200** short listing

Prev Page 1 2 3 [illegible] 6 7 8 9 10 Next Page